Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Roman Gorišek

Izdelava urejevalnika izvorne kode v oblaku z uporabo tehnologij HTML5

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Sodobno računalništvo se vse bolj seli na splet in s tem postajajo spletne aplikacije vse bolj obsežne in kompleksne. Razvoju aplikacij pa mora slediti tudi razvoj programskih jezikov in spletnih standardov. Enega od teh predstavlja peta različica jezika HTML. V okviru diplomskega dela najprej predstavite danes najbolj uporabljene spletne jezike: PHP, javascript in predvsem HTML ter njihov namen. Predstavite napredek, ki ga prinaša HTML5 v primerjavi z njegovim predhodnikom in primerjajte dobre in slabe lastnosti pri prehodu nanj. Izdelajte spletno aplikacijo za urejanje lokalnih datotek s programsko kodo in pregled slik ter podrobno predstavite novosti, ki jih boste uporabili pri razvoju aplikacije.

Izjava o avtorstvu diplomskega dela

Spodaj podpisan Roman Gorišek, z vpisno številko **63060481**, sem avtor diplomskega dela z naslovom:

Izdelava urejevalnika izvorne kode v oblaku z uporabo tehnologij HTML5

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 26. januarja 2015 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

HTML (Hypertext Markup Language) - Označevalni jezik, ki ga brskalnik prikaže kot spletno stran
CSS (Cascading Style Sheets) - Kaskadne stilske podloge, preprost mehanizem za dodajanje stilov, barv, ozadij spletnih stranem
PHP (Hypertext Preprocessor) - Odprto kodni programski jezik, ki se uporablja za strežniške uporabe oziroma za razvoj dinamičnih spletnih vsebin
WWW (World Wide Web) - Splet, svetovni splet je porazdeljen hipertekstni (nadbesedilni) sistem, ki deluje v medmrežju.
Ajax (Asynchronous JavaScript And XML) - Skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij.
XML (Extensible Markup Language) - Preprost računalniški jezik podoben HTML-ju, ki nam omogoča format za opisovanje strukturiranih podatkov ali arhitektura za prenos podatkov in njihovo izmenjavo med več omrežji
ASP (Active Server Pages) - Microsoftov prvi strežniški skriptni jezik za dinamično generiranje spletnih strani.
API (Application Programming Interface) - Skupek rutin, protokolov in orodij za gradnjo programskih aplikacij.
IDE (Integrated Development Envirement) - Aplikacija, ki programerjem ponuja obširne funkcionalniosti za razvoj programskih rešitev.
FTP (File Transfer Protocol) - Standardni omrežni protokol, ki se uporablja za prenos datotek med gostitelji omrežja.
SSH (Secure Shell) - Varen protokol za upravljanje računalnika na daljavo,

ki odjemalcu pošlje geslo v kriptirani obliki.

SSD (Solid-State Drive) - Solid-State Drive je nov tip trdega diska, ki namesto vrtljivih magnetnih plošč, uporablja flash spomin.

CERN (European Organization for Nuclear Research) - Evropska raziskovalna organizacija, ki opravlja z največjim laboratorijem za fiziko delcev na svetu.

NCSA (National Center for Supercomputing Applications) - Ameriško združenje za razvoj in in postavitev kibernetične infrastrukture.

Povzetek

Namen diplomskega dela je predstaviti izdelavo spletne aplikacije za urejanje programske kode in pregled slik. Glavna funkcionalnost aplikacije je urejanje programske kode v spletnem brskalniku, brez nameščanja dodatne programske opreme. Samodejno shranjevanje trenutno aktualnega projekta uporabnikom omogoča neopazen prenos dela iz ene naprave na drugo. Aplikacija uporabnikom omogoča odpiranje datotek ali projektov na lokalnem disku, pregled projekta znotraj aplikacije s pomočjo drevesne strukture, odpiranje več datotek v zavihkih, urejanje besedila, iskanje niza v trenutno odpri datoteki ali znotraj več datotek v projektu in shranjevanje datotek na disk. V aplikaciji je podprto barvanje kode za osem zelo razširjenih programskih jezikov, kar uporabniku olajša branje in urejanje programske kode.

Ključne besede: urejevalnik izvorne kode, spletna aplikacija, splet, strežnik, www, php, javascript, html, html5, oblak.

Abstract

In this thesis we built a source code editor web application and documented all the steps. Main advantage of the application is text editing inside a web browser, with no need of any extra software installation. Automatic server side project saving allows users to change devices seemingly during work. With our application users have a tree structured representation of opened project, can open locally stored files or projects, edit multiple files in tabs, edit text, search for queries in an open file or within a collection of files and also save files back to hard drive. With syntax coloring support for eight common programming languages application makes it easier for users to read, write or edit code.

Keywords: source code editor, web aplication, web, server, www, php, javascript, html, html5, cloud.

Poglavje 1

Uvod

1.1 Opis problema

Nove generacije spletnih tehnologij nam prinašajo vse več možnosti za razvoj zmogljivih aplikacij, ki tečejo na strežniku in uporabniki zanj ne potrebujejo drugega kot spletni brskalnik in povezavo na splet. Ta napredek pa ni zanimiv le pri uporabi aplikacij, temveč tudi pri ustvarjanju novih. Izvajanje aplikacije za razvoj programske opreme na strežniku prinaša številne nove možnosti, kot so deljenje kode s prijatelji, shranjevanje dokumentov na strežniku, uporaba aplikacije iz različnih naprav, tudi mobilnih, samodejno poganjanje in testiranje ustvarjenih spletnih aplikacij in podobno.

Želimo izdelati urejevalnik programske kode v oblaku, ki bo omogočal shranjevanje datotek na lokalni disk. Prav tako bo podprto samodejno shranjevanje datotek na strežnik za registrirane uporabnike. Znotraj aplikacije bo možno tudi pregledovanje slik. Pri doseganju ciljev se bomo osredotočili na orodje HTML5 in si ogledali, katere nove možnosti nam prinaša.

1.2 Pregled obstoječih rešitev

Na spletu najdemo kar nekaj urejevalnikov izvorne kode. Nekateri so zelo preprosti in ponujajo od preprostega okenca za vpis besedila, kot nekakšna

beležka, do specializiranih urejevalnikov, predanih določenemu programskemu jeziku, ki predvsem služijo testiranju manjših odsekov programske kode znotraj brskalnika, vendar ponujajo tudi izvoz kode v datoteko.

Obstaja tudi nekaj rešitev, ki popolnoma delujejo kot sodobni IDEji in poleg urejevalnika izvorne kode ponujajo pregled drevesne strukture projekta, prevajalnike oziroma strežnike za poganjanje kode znotraj brskalnika, razhroščevalnik, barvanje kode in ozadja urejevalnika ... Osredotočili smo se na dve takšni spletni aplikaciji, ki sta brezplačni za uporabo in sta vir inspiracije pri razvoju naše aplikacije. Le-ta bo v prvi različici omogočala le osnovne funkcionalnosti, vendar pa jo bomo razvili z vizijo, da bi v prihodnosti lahko predstavljala programsko opremo, ki bi konkurirala opisanim spletnim aplikacijam.

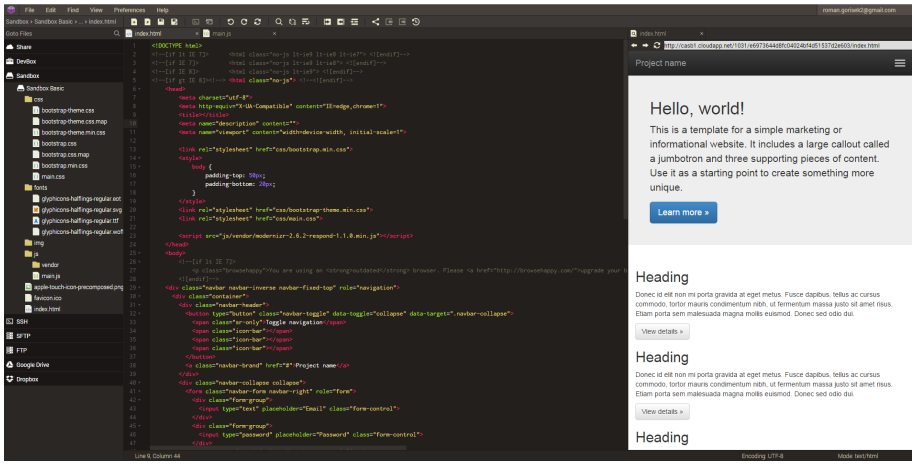
1.2.1 Codeanywhere

Codeanywhere [6] je urejevalnik programske kode v oblaku, ki uporabnikom omogoča razvoj, vzdrževanje in deljenje svojih aplikacij na vseh napravah, po celem svetu. Codeanywhere ponuja brezplačno različico za osnovne uporabnike, lahko pa se odločimo tudi za enega od treh plačljivih paketov, ki omogočijo tehnično podporo, možnost gostovanja več projektov hkrati, dodatne možnosti pri deljenju kode s prijatelji in podobno.

Poleg delovanja znotraj brskalnika obstajajo tudi mobilne različice za operacijski sistem Android in iOS. Spletna aplikacija je posebej koristna, saj uporabnikom omogoča poganjanje kode kar znotraj aplikacije, kar omogoča hitro testiranje. Primer uporabe aplikacije lahko vidimo na Sliki 1.1.

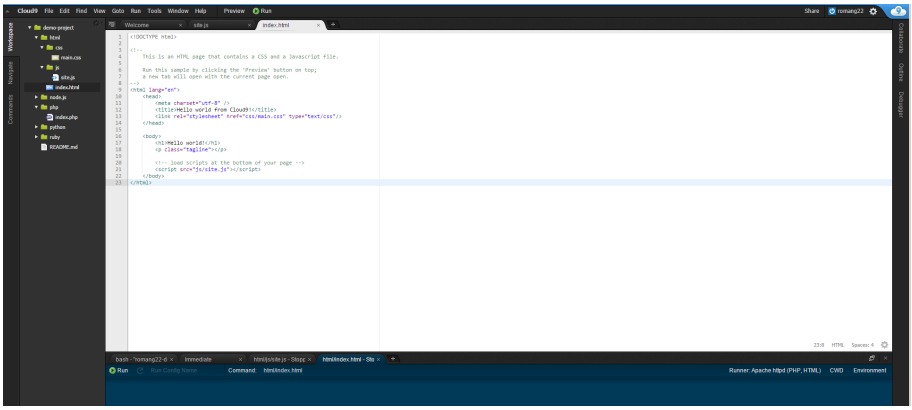
1.2.2 Cloud9

Cloud9 [5] je prav tako urejevalnik izvorne kode v oblaku. Večina osnovnih funkcionalnosti omogoča podobno uporabo kot spletna aplikacija Codeanywhere, nekatere značilne za Cloud9, pa dodajo še dodatne zanimive možnosti za programerje. Ena od teh je možnost deljenega pogleda, ki omogoča pre-



Slika 1.1: Vmesnik aplikacije Codeanywhere.

gledno delo na več datotekah hkrati, ali iskanje sprememb pri posodobljeni datoteki s primerjavo starejše in nove različice. Delovno okolje projekta lahko prav tako delimo s prijatelji, kar omogoča vzporedno urejanje istega dokumenta več oseb. Cloud9 prav tako vsebuje vgrajen nadzor verzij programske kode in še posebej uporabno možnost pregleda sprememb datoteke skozi čas. Vgrajeno testiranje razvitih spletnih aplikacij ponuja ogromno nastavitev, vključno z izbiro kombinacije operacijskega sistema in brskalnika, v katerem bi radi videli končni izdelek. Slika 1.2 prikazuje primer dela na projektu z uporabo urejevalnika Cloud9.



Slika 1.2: Vmesnik aplikacije Cloud9.

Poglavje 2

Tehnologije

V tem delu si poglejmo tehnologije in prosto dostopne knjižnice, ki smo jih uporabili za razvoj svoje aplikacije.

Večina jih predstavlja najpogosteje uporabljene programske jezike in knjižnice pri razvoju spletnih aplikacij, CodeMirror in Downloadify pa sta knjižnici, ki so ju razvile skupnosti spletnih razvijalcev in služijo bolj specifičnim nalogam.

2.1 HTML

Za objavo informacij, ki bodo globalno distribuirane, potrebujemo programski jezik, ki bo razumljiv za vse računalnike po svetu. Jezik za objavo, ki ga uporablja splet, je HTML (Hyper Text Markup Language). Dokument, zapisan v jeziku HTML, je sestavljen iz besedila, ki ga želimo prikazati in iz značk, ki temu besedilu nastavijo, na kakšen način bo prikazan na spletni strani. Značke so ukazi, zapisani med kotnimi oklepaji in predstavljajo različne elemente spletne strani. Poznamo dve vrsti značk, samostojne značke ter začetne in končne značke. Samostojne značke ne potrebujejo zaključka (npr. nova vrstica, nov odstavek ...), saj vsaka naslednja uporaba iste značke prekliče prejšnjo. Včasih pa moramo posebej povedati, do kod seže učinek značke (npr. debeli izpis, centriranje besedila ...). V tem primeru uporabimo končno značko. Končna značka se od začetne značke loči le po doda-

tnem znaku / pred imenom značke. Vse kar se nahaja med začetno in končno značko se podreja oblikovanju, ki ga zahteva značka.

HTML avtorjem omogoča:

- oblikovanje spletne strani s pomočjo elementov, ki so v dokumentu opredeljeni s pomočjo značk,
- objavo spletnih dokumentov z naslovi, besedilom, tabelami, seznami, slikami, ...
- dostop do spletnih vsebin s pomočjo hipertekstnih povezav z enim samim klikom na gumb,
- oblikovanje formularjev za izvrševanje transakcij z oddaljenimi servisi, za uporabo pri iskanju informacij, naročanje produktov, ...
- dodajanje preglednic, video posnetkov, zvočnih posnetkov in ostalih aplikacij neposredno znotraj dokumenta.

2.1.1 Kratka zgodovina HTML

HTML je iznašel Tim Berners-Lee, ko je delal za CERN. Popularen je postal šele po izdelavi brskalnika, ki je bil razvit v NCSA. Po skromnem začetnem obdobju (1990-93) je prišlo nato do eksplozivne rasti svetovnega spleta. Svetovni splet ponuja avtorjem več možnih načinov za prikaz vsebin na spletu. To še spodbuja motivacijo za iskanje skupnih rešitev in standardov.

HTML 2.0 (november 1995) je bil razvit pod pobudo Internet Engineering Task Force (IETF) in se je začel uporabljati konec leta 1994. HTML+ (1993) in HTML 3.0 (1995) sta precej razširila jezik HTML v primerjavi s prejšnjimi različicami. Kljub temu, da ni bil dosežen dogovor o standardizaciji in so obstajale različice jezikov, se je začel zaradi svojih zmožnosti široko uporabljati. Z ustanovitvijo World Wide Web Consortium (W3C) je delovna skupina za HTML uspela najti skupen jezik med raznimi razvijalci in je v

letu 1996 začela standardizacijo HTML. Rezultat je bil HTML 3.2 (januar 1997).

Večina uporabnikov se strinja, da se mora HTML enako prikazovati v različnih brskalnikih in na različnih platformah. To je pa tudi želja razvijalcev, saj na ta način za nižjo ceno izdelajo strani, ker ni treba razvijati različic.

HTML je bil razvit z vizijo, da bo lahko uporabljen na mnogih napravah, ki so priključene na splet: računalnike z različnimi resolucijami in barvnimi globinami, telefone, razne industrijske naprave ...

2.1.2 Sintaksa

Preprost primer dokumenta napisanega v jeziku HTML5:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Title of the document</title>
   </head>

   <body>
      Content of the document......
   </body>
</html>
```

2.2 PHP

PHP se v glavnem uporablja kot skriptni jezik na strani strežnika, saj je posebej dobro prilagojen za ustvarjanje dinamičnih spletnih strani [23, 3]. Med drugim uporabniku ponuja integrirano podporo za dostop do baze podatkov, kot na primer MySQL, zaradi česar je vodilni kandidat za razvoj

najrazličnejših spletnih aplikacij, od preprostih osebnih strani do kompleksnih poslovnih aplikacij.

Spletni brskalniki ne znajo procesirati PHP kode. PHP se procesira na napravi, ki servira dokumente, ki jo imenujemo strežnik (ang. *server*) in šele nato pošlje na uporabnikov brskalnik. PHP nam omogoča ustvarjanje dinamičnih spletnih strani, se pravi strani, ki se glede na pogoje lahko spreminjajo. Na primer, kadar se dve različni osebi vpišeta vsaka s svojim Facebook računom, bosta dobili prikazano različno vsebino, čeprav je naslov (www.facebook.com) isti v obeh primerih. Kaj takega je nemogoče pri dokumentih HTML, ki so statični, kar pomeni, da se ne morejo spremeniti. Vsak uporabnik, ki naloži HTML stran, bo videl natančno isto vsebino.

PHP je tolmačeni programski jezik, kar prav tako predstavlja veliko prednost za PHP razvijalce. Veliko programskih jezikov potrebuje prevajalnik, ki programsko kodo prevede v strojni jezik, ki ga razume računalnik, preden se koda lahko izvede, kar je časovno zahteven postopek. Ker datotek PHP ni treba prevajati, lahko razvijalci urejajo in testirajo svojo kodo veliko hitreje.

2.2.1 Sintaksa

PHP koda je lahko zapisana v obliki preprostih ukazov znotraj kode HTML, lahko pa v samostojni PHP datoteki. Preprost primer kode zapisane v programskem jeziku PHP, ki se nahaja znotraj kode HTML: [20]

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

2.3 JavaScript

JavaScript je programski jezik spleta [14]. Velika večina modernih spletnih strani uporablja JavaScript, vsi moderni brskalniki, na osebnih računalnikih, tabličnih računalnikih in pametnih telefonih, znajo tolmačiti JavaScript, zaradi česar je JavaScript najbolj razširjen programski jezik v zgodovini. JavaScript sodi v skupino treh tehnologij, ki jih mora poznati vsak sodobni spletni programer: HTML za določanje vsebine spletne strani, CSS za določanje predstavitve spletne strani in JavaScript za določanje obnašanja spletne strani. JavaScript je visoko nivojski, dinamičen, tolmačen programski jezik, ki dobro ustreza objektno orientiranemu in funkcijskemu slogu programiranja [1]. Z izjemo površinskih sintaktičnih podobnosti, se JavaScript popolnoma razlikuje od programskega jezika Java, kljub podobnemu imenu. Od začetnih korenin v skriptnemu programiranju se je JavaScript razvil v robusten in učinkovit večnamenski jezik, katerega funkcionalnosti zadostujejo razvoju zahtevnih spletnih aplikacij.

2.3.1 Sintaksa

Preprost primer kode zapisane v programskem jeziku JavaScript, ki se nahaja znotraj kode HTML: [15]

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
```

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph
        changed.";
}
</script>

</body>
</html>
```

2.3.2 jQuery

jQuery je odprtokodna knjižnica, napisana v programskem jeziku JavaScript, ki spletnim programerjem omogoča dodajanje dodatnih funkcionalnosti k njihovim spletnim stranem. V zadnjih letih je jQuery postala najpogosteje uporabljena knjižnica JavaScript pri razvoju spletnih strani. Za uporabo knjižnice jQuery mora uporabnik le vključiti jQueryjev JavaScript dokument znotraj dokumenta HTML. Nekatere spletne strani imajo kopijo knjižnice jQuery dosegljivo na svojem strežniku lahko pa jo uporabniki preprosto naslovijo na enem od gostiteljskih strežnikov, kot je Googlov ali kar jQueryjev. Spletna stran lahko na primer naloži knjižnico jQuery tako, da znotraj glave v dokumentu HTML izvede naslednji klic:

```
<script type="text/javascript"
    src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
```

Ko je knjižnica uspešno naložena, lahko spletna stran uporablja podprtih jQueryjevih funkcij, ki lahko zelo olajšajo naloge, kot so spreminjanje besedila, obdelava podatkov v formularjih, premikanje elementov po strani in izvajanje animacij. jQuery prav tako olajša delo s klici Ajax v sodelovanju s skriptnimi jeziki, kot je PHP ali ASP in tako omogoča komunikacijo s podatkovnimi bazami. jQuery torej ni nadomestek za JavaScript, vendar je nekakšen dodatek, ki omogoča uporabo novih funkcij, ki zmanjšajo količino

kode, ki jo moramo napisati, da dosežemo isti cilj. Ker jQuery teče na strani odjemalca, ne na strežniku, lahko posodablja informacije na spletni strani v realnem času, brez osveževanja strani.

Glavni razlog, da je knjižnica jQuery tako popularna, ni le, da je brezplačna, temveč tudi to, da je skladna z vsemi sodobnimi brskalniki. Ker vsak brskalnik prikazuje HTML, CSS in JavaScript različno, je včasih težko doseči, da je spletna stran videti enako v vseh brskalnikih. Namesto da uporabniki pišejo prilagojene funkcije za vsak brskalnik posebej, lahko uporabijo metodo v knjižnici, ki preverjeno deluje na vseh brskalnikih. Ravno zaradi te podpore so mnogi razvijalci prešli s standardne kode v JavaScriptu na jQuery, da bi si poenostavili proces programiranja [16].

2.3.3 jQuery.tree

jQuery tree je odprtokodna knjižnica, ki s pomočjo knjižnice jQuery močno olajša integracijo datotečne drevesne strukture znotraj spletne strani. jQuery tree omogoča gradnjo drevesa in nato preprosto premikanje po drevesu in odpiranje ter zapiranje map v drevesni strukturi [19].

2.3.4 Ajax

Ajax je kombinacija tehnik spletnega razvoja za ustvarjanje dinamičnih spletnih strani. Kratica Ajax v angleščini pomeni "Asynchronous JavaScript and XML" oziroma asinhroni JavaScript in XML. To pomeni, da spletne strani, ki uporabljajo Ajax, združujejo JavaScript in XML za prikaz dinamične vsebine.

Beseda *asinhroni* se pri Ajaxu nanaša na način, kako se zgodijo klici spletnega strežnika. Ko neka skripta naredi klic na strežnik, ji ta vrne podatke, ki so nato lahko prikazani na spletni strani. Ker se ti koraki zgodijo ob različnih časih, jih obravnavamo kot asinhrone. Večino implementacij Ajaxa uporablja XMLHttpRequst API, ki vsebuje seznam strežniških klicev, ki jih lahko kličemo znotraj JavaScript kode. Strežnik podatke po navadi vrne brskal-

niku v XML obliki, saj jo je preprosto razčleniti, možno pa je podatke vrniti tudi kako drugače oblikovane oziroma neoblikovane, se pravi kot navadno besedilo.

Zmožnost, da skripte tečejo na strani odjemalca ne na strežniku, daje Ajaxu veliko moč. To pomeni, da lahko funkcija, zapisana v jeziku JavaScript, naredi klic na strežnik še po tem, ko je spletna stran v celoti naložena, podatki, prejeti s strežnika, pa so lahko prikazani na strani, ne da bi se morala preostala vsebina ponovno naložiti. Če bi v istem primeru uporabili le skriptni jezik na strežniku, kot je PHP ali ASP, bi se morala osvežiti celotna spletna stran, za prikaz vsake nove vsebine.

S pomočjo Ajaxa je splet postal bolj dinamičen, saj omogoča spletnim stranem, da prejmejo in prikažejo nove vsebine brez ponovnega nalaganja preostanka strani. Z uporabo Ajaxa lahko spletni razvijalci ustvarjajo interaktivne strani, ki učinkovito porabljajo vire in obiskovalcem ponujajo odzivno izkušnjo [4].

2.4 CSS

CSS ali Cascading Style Sheets je značilnost HTMLja, razvita pri W3C. S CSS je možno ustvariti predloge za sloge, ki določajo, kako so različni elementi besedila, odstavki, naslovi, pisave, hiperpovezave in podobno videti na spletni strani. Najpogosteje se CSS uporablja v spletnih straneh napisanih v HTML in XHTML spletnem jeziku, lahko pa se uporablja tudi v poljubnem XML dokumentu. Glavni namen CSSa je, da se vsebinske podatke dokumenta (besedilo, slike, tabele, drugi elementi) loči od podatkov, ki določajo oblikovne lastnosti dokumenta in njegovih elementov. Z uporabo CSSa lahko elementom HTML določimo razne vrste oblikovnih lastnosti, npr. ozadje, barvo, pisavo, odmike, obrobo, položaj, velikost, stopnjo prosojnosti ... [17]

Ločitev vsebine od oblike je namenjena temu, da:

- se zagotovi večja prilagodljivost in nadzor nad specifikacijo lastnosti,
- se zmanjša kompleksnost strani,
- lahko več strani uporablja iste oblikovne lastnosti, ne da bi bilo treba te lastnosti posebej pripisati vsaki strani,
- lahko posamezno stran prikažemo v različnih oblikah (na primer za prikaz na celem zaslonu in za prikaz v oknu).

Delovanje CSSa omogočajo spletni brskalniki, ki vsebujejo CSS podporo. Žal pa spletni brskalniki nekaterih CSS pravil ne tolmačijo identično. Posledica tega je, da je lahko spletna stran v različnih brskalnikih videti različno. Da bi težavo odpravili, oblikovalci uporabljajo CSS filtre, ali pa se izognejo CSS določilom, ki so v različnih brskalnikih različno tolmačeni.

CSS določila lahko vpeljemo v dokument HTML na 3 načine:

- v zunanji stilski datoteki, namenjeni zgolj css kodi. V dokumentu HTML posameznemu elementu pripišemo razred CSS, katerega lastnosti definiramo v poljubni datoteki CSS (npr. style.css),
- v dokumentu HTML v glavi. Podobno kot v prvem primeru, le da ne uporabimo dodatne datoteke,
- kot stilski atribut elementov HTML. V dokumentu HTML posameznemu elementu v ukazu *style* določimo, kakšne lastnosti naj ima.

Posamezen dokument HTML lahko uporablja eno ali več zunanjih stilskih datotek. V primeru, da so v njih za neki dokument na strani določena različna pravila, se vzame pravilo, ki je nazadnje določeno, ali pa pravilo, ob katerem piše !important.

2.4.1 Različice CSSa

CSS je stalno v razvoju. Prva verzija CSS iz leta 1996 je CSS 1, ki je nudila podporo za določanje oblike črk, barve besedil, ozadij in drugih elementov, poravnavo besedil, slik in drugih elementov, razmike, obrobe in položaje večine elementov, identifikatorje elementov in razrede.

Različica CSS 2 je bila objavljena leta 1998 in je nadgradila CSS 1 z novimi lastnostmi, kot so relativno, absolutno in fiksirano postavljanje elementov in večslojnost elementov z uporabo z-index. Omogoča tudi, da črkam določimo senco.

Specifikacija različice CSS 3 je razdeljena v več ločenih dokumentov, imenovanih moduli. Nekaj modulov: Color, Selectors, Namespaces. V modulih so nekatere lastnosti iz CSS 2 nadgrajene, dodane pa so tudi nove.

2.4.2 Sintaksa

CSS sintaksa je preprosta. Konkretni primer iz praktičnega dela:

V HTML dokumentu, v katerem želimo uporabiti zunanjo CSS datoteko, uporabimo ukaz, s katerim uvozimo datoteko "main.css":

```
<link rel="stylesheet" type="text/css" href="pc_css/main.css"
    media="screen" />
```

Datoteka CSS "main.css":

```
/* basic css for the page */

html, body {
   margin: 0px;
   height: 100%;
   font-family: Helvetica, sans-serif;
   color: #000;
   overflow: hidden;
   background-color: #E8E8DA;
```

```
}

#editor_and_file_tabs{
    overflow: hidden;
    background: #0A092C;
    height: 100%;
}
```

2.5 CodeMirror

CodeMirror je vsestranski, odprtokodni urejevalnik izvorne kode napisan v programskem jeziku JavaScript za uporabo v brskalniku. Specializiran je za urejanje programske kode in vsebuje podporo za številne programske jezike in dodatke, ki omogočajo izvedbo bolj zahtevnih funkcij urejevalnika. Bogat programski API in sistem barvanja s pomočjo CSSa omogoča preprosto integracijo znotraj spletne strani in tako dodajanje dodatnih funkcionalnosti [7].

2.6 Downloadify

Je majhna odprtokodna knjižnica, napisana v jeziku JavaScript in Flash, ki omogoča ustvarjanje besedilnih datotek sproti, znotraj brskalnika, brez komunikacije s strežnikom. Knjižnica tako pospeši prenos datoteke (ker ni komunikacije s strežnikom) in zmanjša pretok prometa na strežniku, kot tudi doda zelo uporabno funkcijo “Shrani kot”, ki omogoča izbiro mesta na trdem disku, kjer se nova datoteka shrani [18].

Poglavje 3

HTML5

Ko smo v prejšnjem poglavju predstavili tehnologije, ki smo jih uporabljali za razvoj aplikacije, smo že omenili programski jezik HTML, pri katerem s pomočjo značk določimo strukturo spletnih strani. Ker pa smo se za doseganje želenih rezultatov v večji meri zanašali na funkcije, ki so nove pri HTML5, si bomo najnovejšo različico jezika HTML ogledali bolj podrobno.

HTML5 prinaša številne novosti, ki so ključnega pomena pri razvoju modernih spletnih aplikacij, prav tako pa definira veliko lastnosti, ki so v uporabi že od prejšnjih verzij, vendar niso bile nikoli vključene v katerega izmed standardov. Kot starejše verzije HTML tudi HTML5 ostaja platformno neodvisen, zato predstavlja dobro rešitev za razvoj aplikacij, ki delujejo na vseh operacijskih sistemih s pomočjo ene od novejših različic spletnih brskalnikov (Chrome, Firefox, Opera, Safari, ...). Dobro ga podpirajo tudi mobilne različice spletnih brskalnikov, zaradi česar je HTML5 potencialni kandidat za razvoj platformno neodvisnih mobilnih aplikacij. Veliko zmožnosti standarda HTML5 je bilo razvitih tako, da omogočajo delovanje na manj zmogljivih napravah, kot so pametni telefoni in tablični računalniki [11].

3.1 Novosti

Najbolj zanimiva sprememba, ki jo prinaša HTML5, so brez dvoma nove značke in funkcije ter HTML5 APIji, ki jih lahko uporabljamo pri razvoju spletnih strani. S pomočjo značk lahko na spletno stran dodajamo nove elemente, ki jih do zdaj nismo mogli, kot je na primer video in avdio ali polje za prikaz 2D oblik in grafik. Omogočajo nam boljše strukturiranje dokumenta v smislu semantičnega spleta, saj nove značke bolje opisujejo, kakšen je tip vsebine v njih. V nekaterih primerih pa so le starim značkam dodane razširjene možnosti, tako da so bolj prijazne za uporabo in nam prihranijo delo, kot na primer nove možnosti pri vnosnih poljih. S pomočjo APIjev lahko spletnim aplikacijam dodamo možnosti, kot so geolociranje, več nitno delovanje, delovanje v načinu brez povezave in podobno.

Najbolj zanimivi novi elementi so:

- semantični elementi, kot na primer <header>, <footer>, <artice> in <section>,
- tipi vnosnih polj, kot na primer *color*, *date*, *email* in *range*,
- grafični elementi, kot na primer <svg> in <canvas>,
- multimedijski elementi, kot na primer <audio> in <video>.

Izmed številnih novih HTML5 APIjev (angl. Application Programming Interfaces) si oglejmo nekaj najzanimivejših:

- geolokacija (angl. geolocation) (omogoča pridobivanje geografičnega položaja uporabnika),
- povleci in spusti (angl. drag and drop) (omogoča premikanje objektov HTML),

- lokalno shranjevanje (angl. local storage) (omogoča shranjevanje podatkov lokalno, znotraj uporabnikovega brskalnika),
- pomnjenje aplikacije (angl. application cache) (omogoča shranjevanje spletne aplikacije v pomnilnik in dostopanje do nje tudi brez povezave na internet),
- spletni delavci (angl. web workers) (omogoča izvajanje skript JavaScript v ozadju brez vpliva na odzivnost strani).

Novosti HTML5 pa niso le novi elementi, ki so bili dodani, vendar tudi elementi, ki jih posledično ne potrebujemo več. To seveda ne pomeni, da izgubimo pri možnostih za izdelavo spletnih strani, saj lahko enak učinek dosežemo s pomočjo drugih elementov ali tehnologij, nekaj pa je elementov, ki se niso izkazali kot pomembni, zato njihovega nadomestka ne potrebujemo. V Tabeli 3.1 je prikazano nekaj elementov, ki jih HTML5 ne vsebuje več in novosti, ki omogočajo doseganje istih učinkov.

3.2 Prednosti

HTML5 nam omogoča pisanje bolj čiste, razumljive kode, ki hkrati pripomore k razvoju semantičnega spleta. Z uporabo novih semantičnih elementov namesto elementov *div* je hitro razvidno, kje je glava, noga, navigacija in podobno na strani. Prav tako je z uporabo HTML5 elementov struktura spletnih strani bolj konsistentna, kar pomeni, da programerji in oblikovalci lažje razumejo, kako je stran strukturirana. Veliko funkcij HTML5 je odziv na hitro razvijajočo industrijo mobilnega računalništva, saj omogočajo razvoj spletnih aplikacij, ki so dostopne tako na namiznih računalnikih, kot tudi na prenosnih napravah kot so mobilni telefoni in tablični računalniki. Tu do izraza pridejo možnosti kot na primer lokalno shranjevanje za dostop brez povezave na internet in geolociranje za beleženje in delo z uporabnikovo trenutno lokacijo.

Element	**Nadomestilo**
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	*ni nadomestila*
<frameset>	*ni nadomestila*
<noframes>	*ni nadomestila*
<strike>	CSS
<tt>	CSS

Tabela 3.1: Tabela elementov, ki so v HTML5 odstranjeni

3.3 Pomanjkljivosti

Največja težava pri uporabi HTML5 je ta, da je podprt samo v novejših brskalnikih, kar lahko pomeni, da spletna aplikacija, razvita z uporabo HTML5, morda ne bo pravilno delovala pri uporabnikih, ki imajo starejše različice brskalnikov. Kljub temu, da je programski jezik v večjem delu zelo stabilen, so še vedno nekateri deli, ki se razvijajo oziroma spreminjajo in zato morajo biti razvijalci pozorni, da svoje aplikacije po potrebi posodobijo. Zaradi tega je morda najboljša praksa izdelava aplikacij, ki sicer delujejo v starejših brskalnikih, vendar ponujajo boljšo uporabniško izkušnjo za tiste uporabnike, ki uporabljajo brskalnike, ki podpirajo HTML5. Problematični so lahko tudi multimedijski elementi, saj zaradi težav z licencami vsi formati medijskih datotek niso podprti v vseh brskalnikih. Tako najbolj varno izbiro predstavljata formata MP3 za avdio in MP4 za video, ki sta že podprta v vseh glavnih brskalnikih (Internet Explorer, Chrome, Firefox, Safari in Opera). Seveda pa navedene pomanjkljivosti postajajo vse majn prisotne z vsakim izidom nove

različice popularnih brskalnikov, ki se hitro prilagajajo na HTML5.

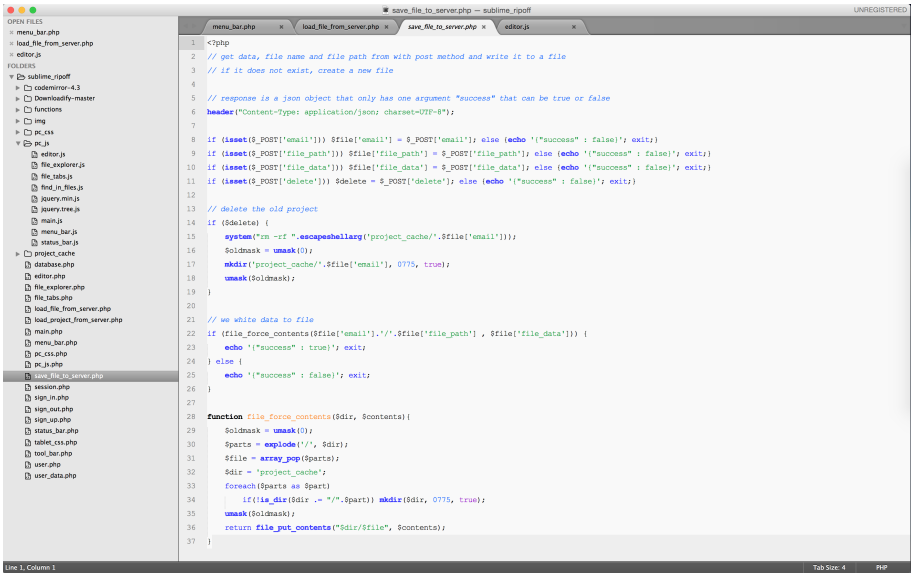
Poglavje 4

Orodja

V tem poglavju si bomo ogledali orodja, ki smo jih potrebovali za razvoj aplikacije. Večina jih je brezplačno dostopnih na spletu, za storitve ponudnika virtualnega zasebnega strežnika DigitalOcean pa je potrebno plačevati mesečno naročnino, vendar so nam na voljo tudi brezplačne alternative. Za storitve DigitalOceana smo se odločili, ker smo jih upobljali že pri prejšnjih projektih in je to pomenilo, da nam ni bilo treba nameščati dodatne programske opreme na računalniku.

4.1 Sublime Text

Sublime Text je platformno neodvisen urejevalnik izvorne kode, ki poleg standardnih vgrajenih funkcij, ponuja tudi razširitve s pomočjo vtičnikov. Večina razširitev je prosto dostopnih ter jih razvija in vzdržuje skupščina uporabnikov. Sublime Text podpira številne programske jezike in programerjem tudi ponuja veliko zelo uporabnih funkcionalnosti, kot so samodejno dokončevanje kode, poganjanje kode znotraj urejevalnika za nekatere jezike, samodejno shranjevanje, istočasno urejanje kode na več mestih,... [24] Aplikacijo Sublime Text vidimo na Sliki 4.1.



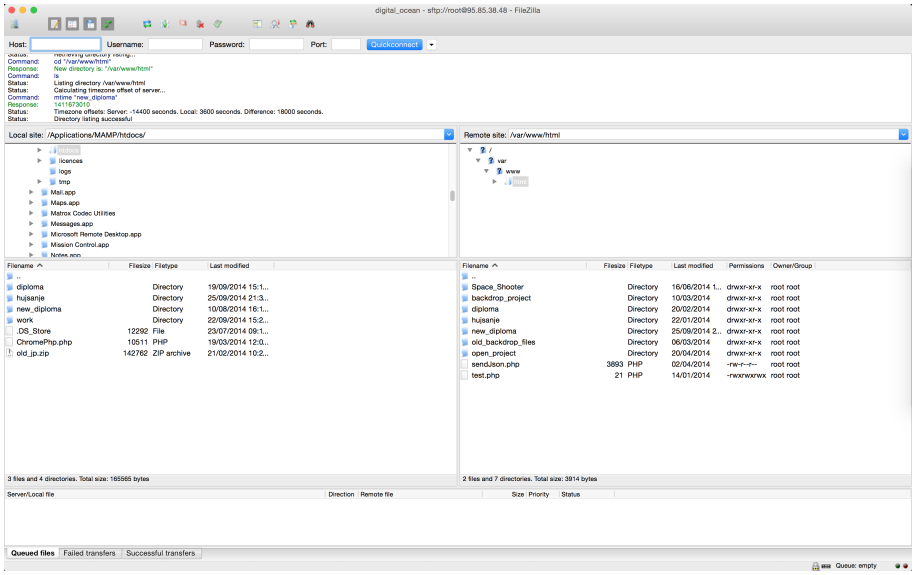
Slika 4.1: Vmesnik aplikacije Sublime Text.

4.2 FileZilla

FileZilla je prosta, platformno neodvisna FTP aplikacija, ki jo sestavljata "FileZilla Client" in "FileZilla Server". Pri delu na nalogi smo uporabljali "FileZilla Client" za prenašanje datotek med našim računalnikom in strežnikom, kjer gostuje naša spletna aplikacija [9]. Aplikacijo FileZilla vidimo na Sliki 4.2.

4.3 PuTTY

PuTTY je aplikacija, ki emulira terminal in deluje kot odjemalec za SSH, Telnet in rlogin protokol. Sprva je bil razvit za Microsoft Windows platformo, kasneje pa so ga prilagodili tudi za druge operacijske sisteme, tudi neuradna prilagoditev za mobilne operacijske sisteme [22]. S pomočjo aplikacije PuTTY smo lahko dostopali do strežnika, na katerem je tekel operacijski sistem Linux, preko terminalskega načina. To nam je omogočalo nameščanje programske opreme in delo z datotečnim sistemom na strežniku.



Slika 4.2: Vmesnik aplikacije FileZilla.

4.4 Google Chrome

Google Chrome je brezplačen spletni brskalnik, ki ga razvija Google. Za izrisovanje spletnih strani se uporablja odprtokodna tehnologija WebKit. Različica beta na operacijskem sistemu MS Windows je bila izdana 2. septembra 2008 v 43 jezikih (vključno s slovenščino), zdaj jih šteje že 50. Prvo stabilno različico so izdali 11. decembra 2008 (1.0.154.36). Različici za Mac OS X in Linux sta bili predstavljeni kasneje. Marca leta 2012 je Chrome prvič postal najbolj uporabljen spletni brskalnik na svetu po raziskavah, ki so jih opravili v analitski hiši StatCounter [10].

4.5 DigitalOcean

DigitalOcean je ponudnik virtualnega zasebnega strežnika, ki izhaja iz New Yorka, vendar ima v najemu strežniške kapacitete po celem svetu (New York, San Francisco, London, Amsterdam, Singapore). DigitalOcean se lahko pohvali z zelo hitro postavitvijo virtualnega strežnika in zagotavljajo uporabo

SSD trdih diskov, kar zagotavlja hitro izvajanje zahtev na strežniku [8].

Poglavje 5

Opis funkcionalnosti aplikacije

Tu bomo predstavili uporabniški vmesnik aplikacije, ki smo jo razvili. Prikazali in opisali bomo glavna orodja, ki jih aplikacija ponuja in podprli s slikami konkretne uporabe aplikacije. Na koncu bomo še poudarili nekaj pomanjkljivosti, na katere smo naleteli med testiranjem.

5.1 Uporabniški vmesnik

Uporabniški vmesnik je zgrajen iz štirih glavnih komponent.

Menijska vrstica, kot smo vajeni pri klasičnih namiznih aplikacijah, ponuja vrsto menijev, ki vsebujejo funkcije za delo s projektom oziroma posameznimi datotekami (odpri, zapri, shrani, nova datoteka ...), funkcije za urejanje programske kode znotraj urejevalnika (razveljavi, zakomentiraj/odkomentiraj, kopiraj, prilepi, najdi ...) in funkcije za prirejanje uporabniškega vmesnika osebnim preferencam (prikaži/skrij komponento, barvanje programske kode, barvanje urejevalnika, velikost pisave ...).

Stranska vrstica prikazuje drevesno datotečno strukturo projekta, ki je trenutno odprt in nam ob pregledu datotek omogoča odpiranje in zapiranje map ter odpiranje datotek znotraj urejevalnika z enim samim klikom.

Statusna vrstica služi za prikaz obvestil uporabniku in vsebuje splošne informacije o urejevalniku, kot je trenutni položaj kurzorja, velikost vstavlje-

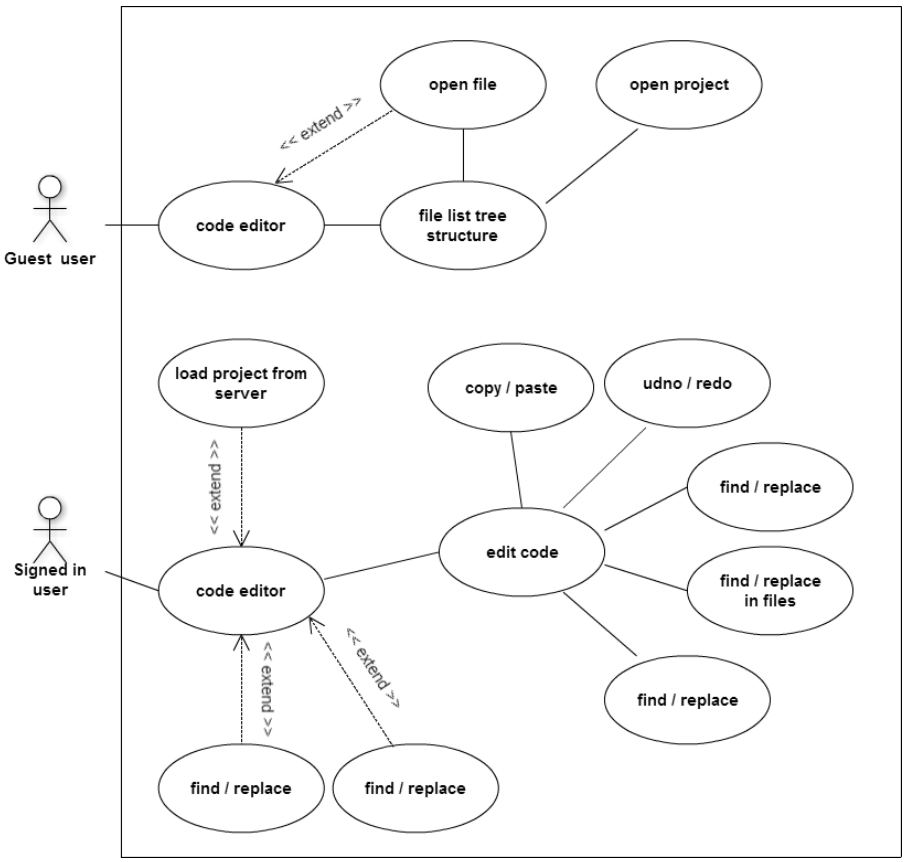
nega presledka ob pritisku tabulatorja in trenutni izbran programski jezik za barvanje sintakse programske kode.

Zadnja in glavna komponenta je seveda urejevalnik izvorne kode, ki ga v veliki večini sestavlja okno za vnos besedila, kjer lahko urejamo izvorno kodo, ob levi stani pa je pas, kjer so oštevilčene vrstice kode, kar uporabnikom olajša iskanje napak in podobno. Takoj nad oknom za vnos besedila je vrstica zavihkov, ki ponazarjajo trenutno odprte datoteke in nam omogoča hitro menjavo med njimi in nazorno prikazuje zavihek datoteke, ki je trenutno prikazana v urejevalniku. Če je število odprtih datotek preveliko za prikaz vseh zavihkov, se na koncu vrstice pojavi gumb za odpiranje spustnega menija z zavihki, ki jih ni bilo mogoče prikazati.

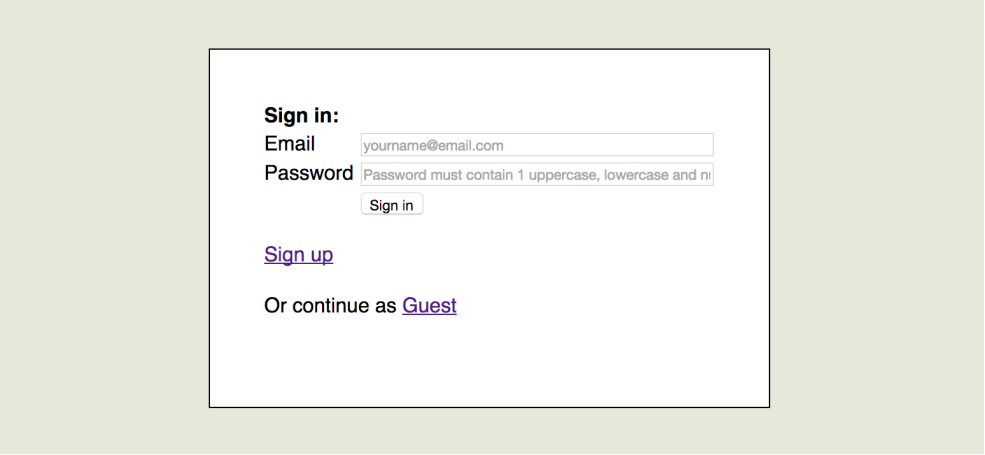
5.2 Prikaz uporabe

Na diagramu uporabe, ki ga prikazuje Slika 5.1, vidimo primer uporabe aplikacije za registriranega uporabnika, tako kot tudi za neregistriranega. Kot je razvidno, je glavna prednost registriranih uporabnikov ta, da jim projekta ni treba odpreti vedno znova iz lokalnega diska, vendar se kopija shrani na strežniku in ob ponovnem obisku spletne aplikacije tudi samodejno naloži. Na diagramu smo pri neregistriranem uporabniku predstavili primer uporabe, ko uporabnik želi v aplikaciji odpreti nov projekt. Primer uporabe registriranega uporabnika pa prikazuje dodano možnost samodejnega nalaganja projekta in tudi prikazuje druge funkcionalnosti, ki jih ponuja aplikacija in služijo urejanju same programske kode. V nadaljevanju si bomo podrobneje ogledali delovanje nekaterih funkcionalnosti.

Ko se odpre začetna stran spletne aplikacije, se naprej prikaže okno za vpis uporabnika, kot je prikazano na Sliki 5.2. Tu lahko vnesemo uporabniško ime in geslo ali sledimo povezavi na obrazec za dodajanje novega uporabnika, lahko pa tudi nadaljujemo kot gost, kar pomeni, da nam ne bo na voljo samodejno shranjevanje projekta na strežniku.

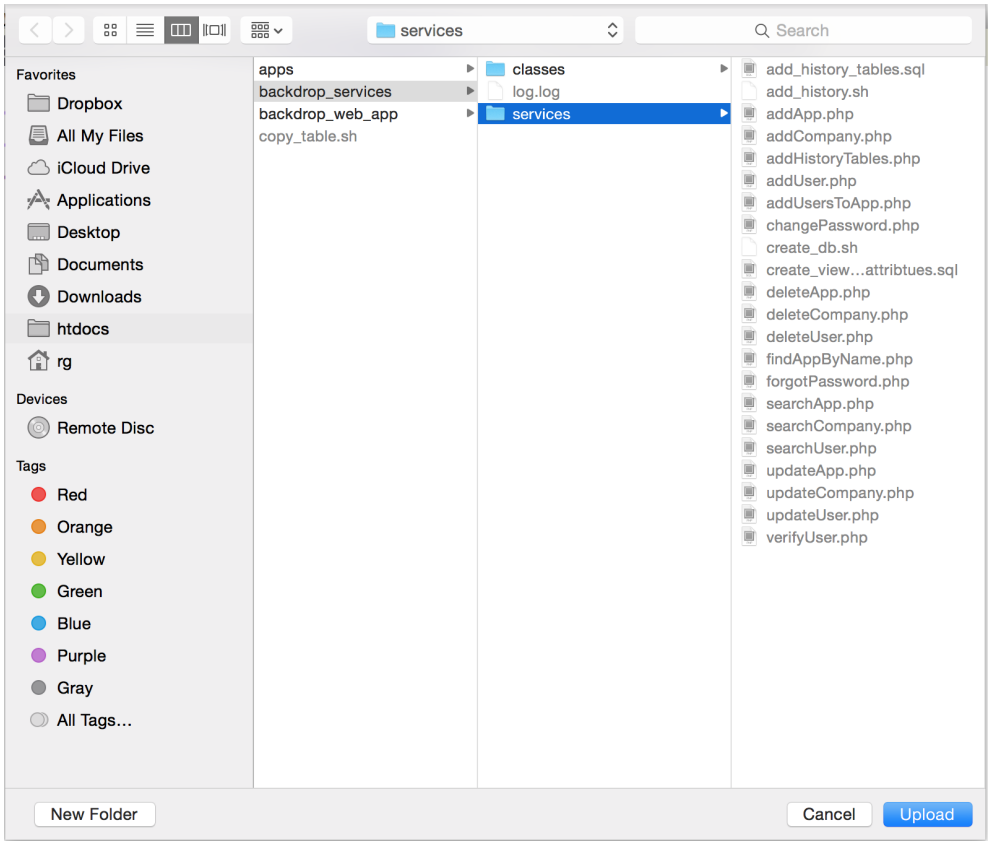


Slika 5.1: Diagram primerov uporabe za registriranega in neregistriranega uporabnika.

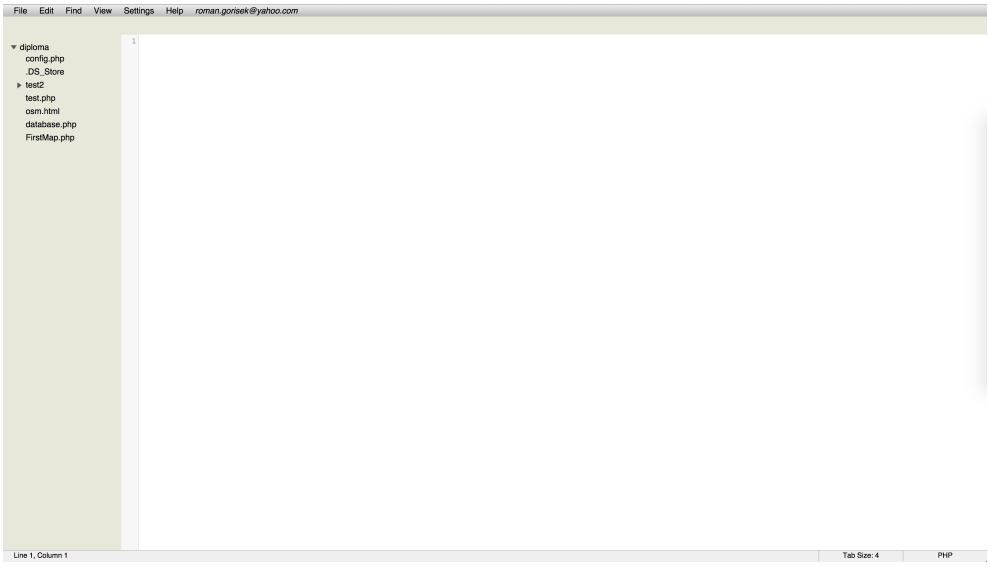


Slika 5.2: Prijavno okno.

Če smo se vpisali kot registrirani uporabnik in smo aplikacijo že uporabljali, se na samodejno naloži zadnji projekt, na katerem smo delali. V nasprotnem primeru se odpre glavno okno aplikacije brez odprtega projekta. Projekt lahko odpremo tako, da v menijski vrstici izberemo "File" in nato "Open...", ki nam odpre okno za izbiro mape, v kateri je projekt, ki ga vidimo na Sliki 5.3. Slika 5.4 prikazuje, kako je videti aplikacija, ko je odprt projekt.

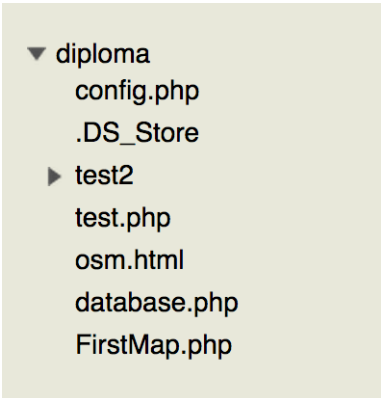


Slika 5.3: Okno za izbiro mape s projektom.



Slika 5.4: Aplikacija z odprtim projektom.

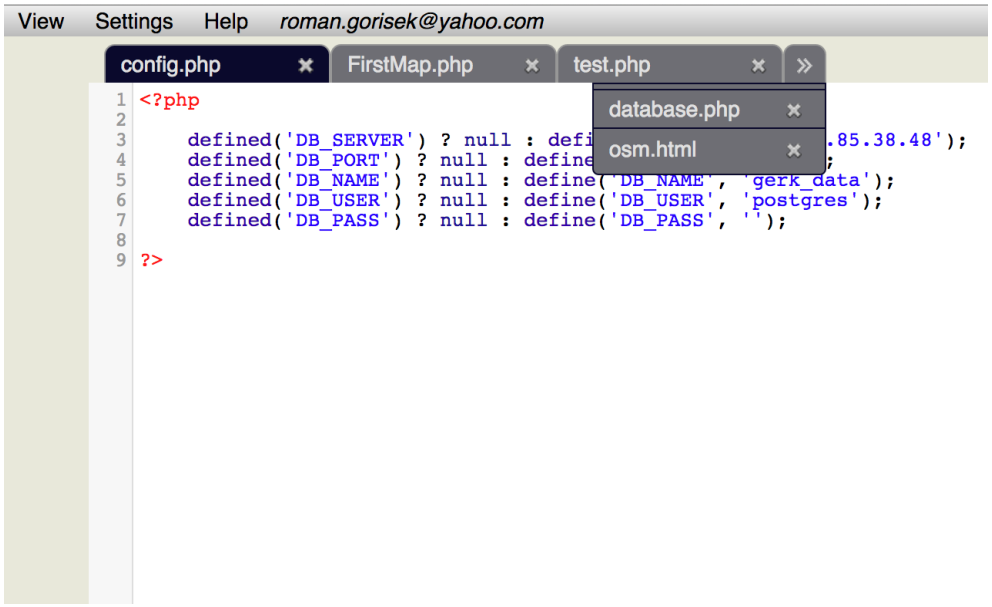
Odprt projekt je viden v drevesni strukturi na levi strani aplikacije. Mape so predstavljene s puščico levo od imena mape. Če puščica kaže levo to pomeni, da je mapa zaprta in jo s klikom na ime lahko odpremo. Poleg odprte mape puščica kaže navzdol. Odprt projekt vidimo na Sliki 5.5.



Slika 5.5: Drevesna struktura odprtega projekta.

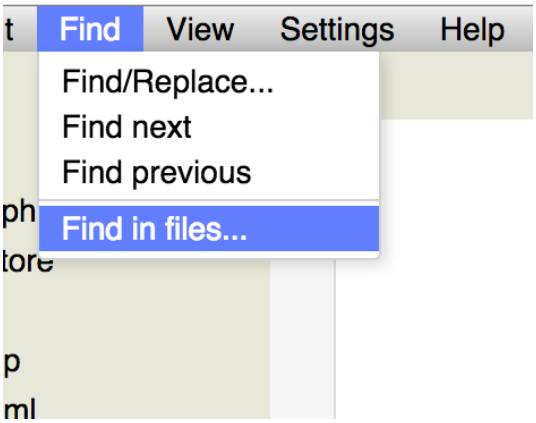
Tudi datoteko odpremo s klikom na njeno ime v drevesni strukturi. To

povzroči, da se besedilo datoteke prikaže v urejevalniku, nad njim pa se odpre nov zavihek z imenom datoteke. Urejevalnik in zavihke vidimo na Sliki 5.6.



Slika 5.6: Datoteka odprta v urejevalniku.

V menijski vrstici lahko najdemo funkcije, ki nam pomagajo pri urejanju datotek. Ena teh je iskanje besedila v več datotekah. Na Sliki 5.7 vidimo, da se funkcija za iskanje v več datotekah nahaja v meniju "Find". Ob kliku na "Find in files..." se v urejevalniku prikaže okno za nastavitve iskanja, prikazano na Sliki 5.8. Ko se iskanje zaključi, pa rezultate vidimo v urejevalniku, kot da bi odprli datoteko z navadnim besedilom. To prikazuje Slika 5.9.



Slika 5.7: Meni “Find” v menijski vrstici.

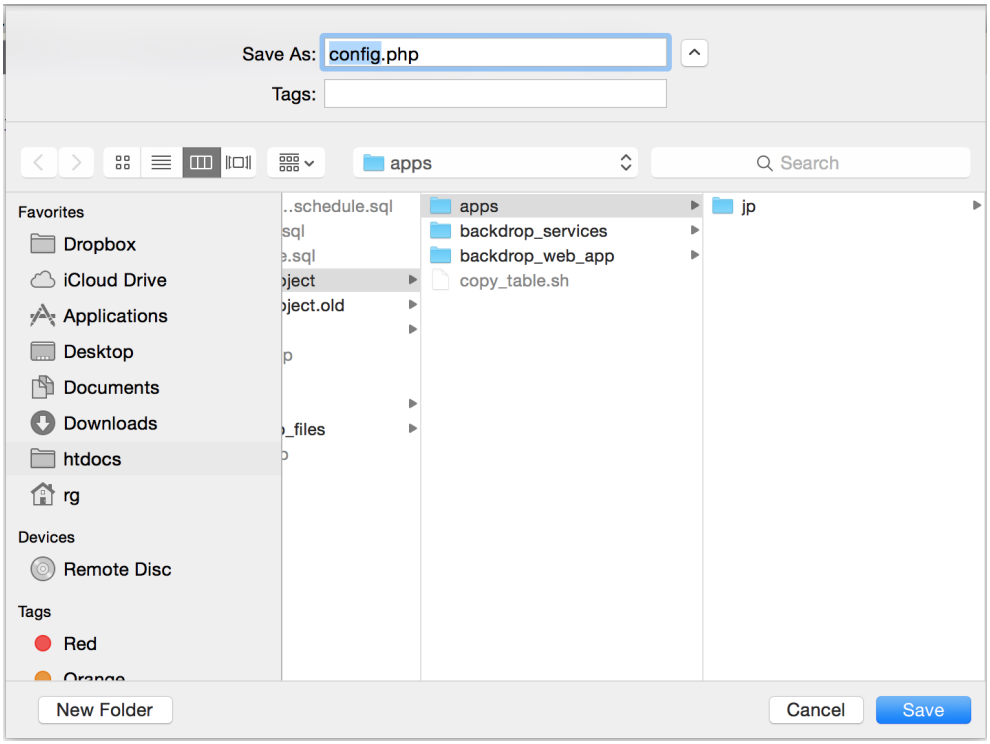
Slika 5.8: Okno z nastavitvami za iskanje.

Find Results

```
Search results for "require":


diploma/test2/test.php:

...<?php

    require_once 'database.php';
...

diploma/test.php:

...<?php

    require_once 'database.php';
...

diploma/database.php:

...<?php
require_once("config.php");

c...

Matches found in file names:


No file names match!
```

Slika 5.9: Rezultati iskanja prikazani v urejevalniku.

Ko smo z urejanjem datoteke končali, jo lahko shranimo na trdi disk na napravi, s katero delamo. S klikom na "Save as..." v meniju "File" odpremo okno za izbiro lokacije, kjer želimo shraniti novo datoteko, preden jo shranimo, pa si lahko izberemo tudi, kako jo bomo poimenovali. Če gre za spremembo obstoječe datoteke, lahko staro različico nadomestimo z novo. Okno za shranjevanje datotek je prikazano na Sliki 5.10.



Slika 5.10: Okno za shranjevanje datotek.

5.3 Pomanjkljivosti in možne izboljšave

Namen naloge je bil implementirati urejevalnik programske kode v oblaku, ki omogoča razvoj aplikacij, pri čemer smo se predvsem osredotočili na razvoj spletnih aplikacij, za kar je tudi podprtih največ programskih jezikov. Svoj cilj smo sicer dosegli, vendar so še vedno prisotne pomanjkljivosti, ki bi jih bilo treba odpraviti, preden bi aplikacija bila zadovoljiva za dejansko uporabo na projektih.

Najprej omenimo funkcijo "Shrani kot" (angl. *Save as*), ki je implementirana in deluje v redu, saj uporabniku ponudi okno za izbiro mape, kjer želi shraniti datoteko, vendar pa to pomeni uporabo tehnologije Adobe Flash. HTML, tudi v zadnji različici HTML5, namreč ne ponuja možnosti dostopa do datotečnega sistema pri odjemalcu. Dostop do datotečnega sistema sicer

lahko predstavlja tveganje pri varnosti aplikacije, vendar "Shrani kot" pomembna funkcionalnost, saj omogoča posodabljanje in ustvarjanje datotek brez ročnega kopiranja.

Verjetno največja pomanjkljivost aplikacije je, da uporabnik, ko shrani novo datoteko nekje znotraj projekta, le-te ne vidi v drevesni strukturi projekta. Projekt je najprej treba ročno ponovno naložiti z uporabo funkcije "Odpri". Do tega ponovno pride zaradi varnostnih razlogov, saj nam JavaScript ne dopušča, da bi programsko odprli datoteko na trdem disku odjemalca.

Aplikacija omogoča predogled slik, ni pa možno slik urejati. To morda ni velika pomanjkljivost, vendar ker je tako imenovano platno (angl. *canvas*) [13] ena večjih prednosti, ki jih prinaša HTML5, bi bilo smiselno dodati to možnost. Zaradi časovne omejitve to v prvi iteraciji aplikacije ni bilo možno, bi pa to bil smiseln naslednji korak.

Med testiranjem uporabe aplikacije smo ugotovili, da nam aplikacija ne prikaže, ali smo datoteko spreminjali in spremembe nismo shranili. Za registrirane uporabnike aplikacija sicer podpira samodejno shranjevanje na strežnik, pri neregistriranih uporabnikih pa se pojavi nevarnost, da zaprejo aplikacijo, preden shranijo svoj napredek.

Poleg popravkov za pomanjkljivosti, ki smo jih našteli, je kot po navadi pri razvoju aplikacij, veliko prostora za izboljšave. Poglejmo si nekaj najbolj očitnih in takšnih, ki bi lahko najbolj povečale uporabnost aplikacije. Najprej bi verjetno morali dodati podporo za barvanje kode za čim več programskih jezikov, kar je ključnega pomena pri delu s programsko kodo in predstavlja eno najosnovnejših funkcionalnosti naše aplikacije. S tem bi zagotovili, da bi naša aplikacija lahko nadomestila vse več plačljivih aplikacij in aplikacij odvisnih od platforme in naprave.

Še ena stvar, ki morda odvrača uporabnike od uporabe naše aplikacije, je potrebna povezava na splet. Precej elegantna rešitev za to je shranjevanje aplikacije v predpomnilnik (angl. *aplication cache*) [12], ki je ena od novosti

pri HTML5. S pomočjo le-tega si brskalnik spletno aplikacijo shrani, kar prinese naslednje prednosti:

- dostop do spletne strani brez povezave na internet,
- hitrejše delovanje, saj se dokumenti naložijo prej, če so shranjeni v brskalniku in jih ni treba prenašati iz strežnika,
- zmanjšana obremenitev strežnika, saj se prenesejo le dokumenti, ki so bili posodobljeni ali so novi.

Glede na to, da naša aplikacija deluje v oblaku, bi bilo smiselno dodati tudi možnost deljenja projekta s prijatelji in urejanje dokumenta več oseb hkrati. S tem bi izboljšali možnosti sodelovanja in medsebojne pomoči pri manjših skupinah, ki sodelujejo na projektih.

Za povečanje produktivnosti bi lahko dodali še okno za predogled, ki bi z uporabo namenskih strežnikov poganjal napisano kodo in vračal rezultate uporabniku, tako da le-temu ne bi bilo potrebno preklapljati med aplikacijami oziroma ročno poganjati kode. Če bi se predogled izvajal samodejno v večnitnem načinu in se posodobil, ko bi uporabnik napisal novo kodo, bi bilo še bolje.

Poglavje 6

Implementacija

V tem delu podamo opis izvedbe implementacije sistema. Poglavje je razdeljeno na razvoj na strani strežnika in na strani odjemalca. Opis implementacije orodij smo obogatili s primeri izvorne kode, ki smo jih podrobno opisali.

6.1 Razvoj na strani strežnika

Za razvoj aplikacije na strani strežnika smo uporabljali programski jezik PHP, ki ga v našem primeru najbolj potrebujemo za delo z uporabniškimi računi, shranjevanje kopije projekta na strežniku in vzpostavitev seje. Ko se nov uporabnik prijavi preko spletnega obrazca, se vneseni podatki pošljejo skripti, ki le-te naprej preveri in če so podatki smiselni, se kliče funkcija za dodajanje novega uporabnika. Ta funkcija skrbi za shranjevanje podatkov v zelo preprosto relacijsko bazo, kjer je tabela uporabnikov.

```
// get the data submitted by the form and process it
if (isset($_POST["submit"])) {
   if ($_POST["email"] && $_POST["password"] &&
       $_POST["passwordCheck"]) {
      if (strcmp($_POST["passwordCheck"], $_POST["password"]) == 0)
         {
```

```
        // search for email
        if (!User::email_already_exists($_POST["email"])) {
           // add user
           $user = User::sign_up($_POST["email"],
               $_POST["password"]);
           if (!$user) {
              // error wrong email or password
              echo "whong email or password";
           }
        } else {
           // error user already exists
           echo "user already exists";
        }
     }
  }
}
```

Zgornji odsek kode prikazuje preverjanje podatkov in klic funkcij za delo z uporabniki. *User::email_already_exists($_POST["email"])* je ukaz, ki v razredu *User* kliče funkcijo *email_already_exists* in ji poda elektronski naslov, ki ga je vpisal uporabnik. Funkcija je zelo preprosta. S pomočjo SQL zahteve v bazi poskušamo najti uporabnika, ki ima prejeti elektronski naslov in če poizvedba vrne kakšen vpis bo metoda vrnila *true* sicer pa *false*. Izvajanje SQL zahtev nam omogočajo integrirane funkcije PHPja, ki smo jih zbrali v razredu *PgDatabase*. Poglejmo si funkciji *open_connection()* in *prepare_execute*. Pri prvi uporabimo integrirano funkcijo *pg_connect*, ki vzpostavi povezavo s PostgreSQL bazo, pri drugi pa smo uporabili *pg_prepare*, *pg_execute* in na koncu še *pg_fetch_all*. Tu najprej pripravimo strukturo stavka SQL, nato ga poženemo s podatki, ki jih vsebuje in na koncu še interpretiramo odgovor podatkovne baze, tako da ga lahko uporabimo v PHP kodi. Takšen postopek je priporočljiv, saj preprečuje napad z vrivanjem SQL ukazov.

Izvorna koda funkcij *open_connection()* in *prepare_execute*:

```
public function open_connection() {
   $this->connection = pg_connect("host=".DB_SERVER."
       port=".DB_PORT." dbname=".DB_NAME." user=".DB_USER."
       password=".DB_PASS);

   if (!$this->connection) {
      die("Database connection failed: " .
          pg_last_error($this->connection));
   }
}

public function prepare_execute($sql, $data) {
   $this->last_query = $sql;
   pg_prepare($this->connection, "", $sql) or die
       (pg_last_error($this->connection));
   $result = pg_execute($this->connection, "", $data);
   $resultArray = pg_fetch_all($result);
   return $resultArray ? $resultArray : false;
}
```

Poglejmo si še skripto PHP, ki nam vrne tabelo poti vsake datoteke projekta, ki je shranjen na strežniku:

```
// open a folder with last project of a user and return json array
    with file paths

header("Content-Type: application/json; charset=UTF-8");

if (isset($_POST['user_folder'])) $user_folder =
    $_POST['user_folder']; else { echo '{"success": false, "msg":
    "no user folder was send"}'; exit; }
$dir = 'project_cache/'.$user_folder.'/';

if (!is_dir($dir)){
```

```
   echo '{"success": false, "msg": "no user folder was found for :
       '.$dir.'"}'; exit;
}

echo '{"success": true, "fileList": [';
openDirR($dir, true, 0);
echo ']}';

function openDirR($dir, $first){
   if (is_dir($dir)){
      if ($dh = opendir($dir)){
         while (($file = readdir($dh)) !== false){
            if ($file != '..' && $file != '.') {
               if (is_dir($dir.$file.'/')) {
                  openDirR($dir.$file.'/', $first);
               } else {
                  $dir_array = explode("/", $dir);
                  array_shift($dir_array); array_shift($dir_array);
                  if ($first) $first = false; else echo ', ';
                  echo '"'.implode("/", $dir_array).$file.'"';
               }
            }
         }
         closedir($dh);
      }
   }
}
```

Najprej opazimo nastavitev *header*, s katero smo dosegli, da skripta vrača odgovor v JSON obliki in jo lahko kličemo v JavaScript kodi s pomočjo Ajax klica. Nato preberemo podatke, ki so bili poslani z metodo POST, ko je bila skripta klicana. Tokrat potrebujemo le podatek o imenu mape, ki jo uporabnik uporablja za shranjevanje projekta. Ko se prepričamo, da mapa,

ki bi jo radi prebrali obstaja, sestavimo strukturo JSON. Element *success* nam pove ali, se je skripta izvedla brez napak, element poimenovan *fileList* pa je tabela elementov tipa *string*. Vsak od teh elementov predstavlja eno datoteko v uporabnikovem projektu. Da smo res našli vse datoteke, smo se morali rekurzivno sprehoditi po datotečni strukturi. Ker v PHPju ni takšne funkcije vgrajene, smo napisali funkcijo *openDirR*, ki rekurzivno sestavi poti do datotek.

6.2 Razvoj na strani odjemalca

Za razvoj funkcionalnosti, ki delujejo na strani odjemalca, uporabljamo HTML5 in JavaScript. S pomočjo HTMLja lahko postavimo statično spletno stran, kot nekakšne temelje aplikacije, JavaScript pa uporabimo za dodajanje funkcij, za komunikacijo s strežnikom, za interakcijo z uporabnikom in podobno. Količinsko morda HTML koda predstavlja zanemarljiv del v primerjavi s kodo JavaScript, vendar pa je prav tako pomembna. Pogledali si bomo nekaj konkretnih primerov iz naše aplikacije, kjer smo uporabili nepogrešljive novosti HTML5 v kombinaciji z JavaScriptom in predstavili, kako se ta programska jezika dopolnjujeta.

Za začetek si oglejmo kodo HTML, s katero smo izdelali obrazec za prijavo uporabnika. Glavni gradnik obrazcev so vnosna polja, ki uporabniku omogočajo vnos podatkov. V spodnjem primeru imamo tri različne tipe elementa za vnos, prvi služi za vnos navadnega besedila, konkretno za elektronski naslov, drugi za vnos gesla, ki pri vpisu skrije znake, ki jih vnaša uporabnik in na koncu še element za potrditev, ki deluje kot gumb, ki sproži pošiljanje podatkov. Pri prvih dveh opazimo poleg tipa, imena in velikosti še dodatne nastavitve, ki so nove pri HTML5 in odpravljajo potrebo po uporabi JavaScripta za preverjanje podatkov. Nastavitev *placeholder* nam olajša nastaviti besedilo, ki se znotraj vnosnega polja prikazuje, ko je to sicer prazno in izgine, ko uporabnik začne vnašati svoje podatke. Takšno besedilo služi kot pomoč pri izpolnjevanju obrazcev. Potem je tu nastavitev *requi-*

red, ki nastavi polje kot obvezno, kar pomeni, da uporabnik ne more poslati podatkov, če tega polja ni izpolnil. Na koncu pa je tu še zelo uporabna opcija *pattern*, ki prinaša integrirano podporo za preverjanje vnesenih nizov, z uporabo regularnih izrazov. Tako ne potrebujemo več dodatnih funkcij, ki preverjajo na primer, ali je uporabnik res vpisal elektronski naslov, preden te podatke pošljemo na strežnik.

```
<form action="sign_in.php" method="post" accept-charset="utf-8">
   <table>
      <tr>
         <td><b>Sign in:</b></td><td></td>
      </tr>
      <tr>
         <td><label for="email">Email</label></td><td><input
             type="email" name="email" size="50"
             placeholder="yourname@email.com"
             pattern="^[a-zA-Z][a-zA-Z0-9-_\.@]*$" required /></td>
      </tr>
      <tr>
         <td><label for="password">Password</label></td><td><input
             type="password" name="password" size="50"
             placeholder="Password must contain 1 uppercase,
             lowercase and number"
             pattern="(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$"
             required /></td>
      </tr>
      <tr>
         <td></td><td><input type="submit" name="submit"
             value="Sign in"></td>
      </tr>
      <tr>
         <td colspan="2"><br><a href="sign_up.php">Sign up</a></td>
      </tr>
```

```
      <tr>
         <td colspan="2"><br>Or continue as <a
             href="main.php">Guest</a></td>
      </tr>
   </table>
</form>
```

HTML5 pa ni prinesel le dodatnih možnosti pri vnosnih poljih, vendar tudi nove tipe. Eden teh je tip *file*, ki omogoča, da v brskalnik naložimo eno ali več datotek. To možnost smo izkoristili za odpiranje projekta, ki ga uporabnik želi urejati s pomočjo spletne aplikacije. Za dodajanje gumba za odpiranje datotek je potrebna le preprosta vrstica:

```
<input type="file" id="file_input" webkitdirectory=""
    directory="">Open...</input>
```

Nam pa to odpre veliko možnosti za delo z naloženimi datotekami s pomočjo JavaScripta.

```
$('#file_input').change(function(e) {
        // draw the directory tree
        if (e.target.files.length > 0) {
            tree.init(e)
            // clear the cache of open tabs and close all tabs
            if (localStorage.getItem("open_blobs") != null)
                localStorage.removeItem("open_blobs");
            $('.x_btn').click();
            // save the loaded project to the server if logged in
            if (user.logged) {
                saveToServer(tree.fileList, user.email);
            }
        }
    });
```

V našem primeru uporabili funkcijo, ki se sproži ob spremembi vnosnega polja

za datoteke. V funkciji iz naloženih datotek zgradimo in prikažemo drevesno strukturo, zapremo in izbrišemo iz pomnilnika morebitne odprte datoteke iz prejšnjih projektov in ustvarimo varnostno kopijo projekta na strežniku. Pri brisanju datotek iz pomnilnika vidimo, da smo uporabljali *localStorage*, kar nas pripelje do še ene novosti pri HTML5. S pomočjo funkcije *localStorage* lahko shranjujemo precej velike podatke v brskalniku, ki bodo dostopni vse, dokler se brskalnik ne zapre. Pri aplikaciji to uporabimo za shranjevanje seznama odprtih datotek. Shranjevanje deluje v obliki ključ - vrednost, ki sta vedno shranjena kot navadno besedilo. Zato smo se odločili, da za odprte datoteke ustvarimo JSON objekt, ki ga nato kot navadno besedilo shranimo pod ključ *open_blobs*.

Koda s katero ustvarimo JSON objekt odprtih datotek in ga shranjujemo v *localStorage*:

```
open_blobs[file_id] = {};
open_blobs[file_id]["type"] = file_type;
open_blobs[file_id]["name"] = file.name;
open_blobs[file_id]["url"] = window.URL.createObjectURL(file);
localStorage.setItem("open_blobs", JSON.stringify(open_blobs));

// open file tab
openTab(file.name, file_id);
// open file in editor
var stored_file =
    JSON.parse(localStorage.getItem("open_blobs"))[file_id];
openFile(stored_file.type, file_id, stored_file.url, true);
```

Z uporabo HTML5 nam je na voljo tudi HTML API, imenovan *web worker*, ki nam omogoča poganjanje skript JavaScript v ozadju, kar pomeni, da lahko izvajamo funkcijo in spletno stran med tem še vedno uporabljamo. *Web worker* smo uporabili za iskanje nizov v več datotekah hkrati, saj je to naloga, ki lahko hitro postane zelo časovno zahtevna. Več datotek kot

izberemo in večje kot so datoteke, dalj časa potrebujemo, da preiščemo vse besedilo. Ker pa se skripta izvaja v ozadju, lahko uporabnik aplikacijo neovirano uporablja med tem, ko le-ta išče zadetke.

Poglejmo si odsek kode, kjer uporabimo *web worker*:

```
var w;
if(typeof(Worker) !== "undefined") {
    if(typeof(w) == "undefined") {
        w = new Worker("pc_js/find_in_files.js");
    }
    w.onmessage = function(event) {
        // open the file with results and close worker
        var file_url = window.URL.createObjectURL(event.data);
        console.log(file_url);
        w.terminate();
    };
    w.postMessage(data);
}
```

Kot vidimo, smo naredili *web worker*, ki bo uporabljal skripto *find_in_files.js*, ki se nahaja v mapi *pc_js*. Skripta prejme nastavitve iskanja in nato zaporedno naloži vsako datoteko, ki jo želimo preiskati, si shrani, kje so besede, ki jih iščemo in ko preišče vse datoteke, vrne rezultat iskanja. Za komunikacijo med aplikacijo in skripto uporabimo funkcijo *postMessage*.

Poglavje 7

Sklepne ugotovitve

V diplomski nalogi smo razvili spletno aplikacijo za urejanje programske kode, ki je brezplačno dostopna na spletu in ne potrebuje namestitve. Kljub temu jo lahko uporabimo za delo na lokalnih projektih, saj podpira odpiranje datotek iz diska in shranjevanje nanj.

Cilj naloge je uporabniku zmanjšati količino aplikacij, ki jih potrebuje za razvoj aplikacij in posredno olajšati in pospešiti razvoj. Pospešitev razvoja je lahko precej velika, če uporabnik za razvoj uporablja več naprav (domači računalnik, prenosnik, službeni računalnik), saj lahko menja naprave brez težav. Za registrirane uporabnike se projekt shranjuje na strežniku in se samodejno naloži zadnje stanje, ko se uporabnik vpiše na drugi napravi.

Za razvoj aplikacije smo uporabili odprtokodna orodja PHP, HTML, JavaScript in jQuery ter CSS. Zgradba aplikacije je zelo preprosta, kar pomeni, da jo lahko skaliramo tako vertikalno kot horizontalno. Pri razvoju nismo imeli večjih težav, razen z dostopom do lokalnega diska pri odjemalcu, saj je ta zelo omejen zaradi varnosti. Težavo smo rešili s pomočjo orodja Flash, vendar to predstavlja pomanjkljivost, ki bi jo v prihodnosti radi odpravili.

Večje pomanjkljivosti smo že omenili v prejšnjih poglavjih, vredno pa je še omeniti, da je bila aplikacija razvita in testirana na operacijskem sistemu OSX in v brskalniku Google Chrome. Posledično bi, pred morebitno objavo, aplikacijo morali preizkusili še na drugih operacijskih sistemih in z uporabo

različnih brskalnikov, saj se prikaz lahko precej razlikuje. Poleg tega bi aplikacijo predstavili testni skupini uporabnikov in na podlagi kritik prilagodili uporabniško izkušnjo in funkcionalnosti.

Pri razvoju aplikacije smo se naučili, da je HTML5, skupaj s CSS3, prinesel veliko izboljšav, ki omogočajo izdelavo bolj zmogljivih in modernejših spletnih aplikacij, vendar pa je to še vedno orodje za gradnjo statičnih spletnih strani. Pri razvoju zmogljivejših aplikacij v oblaku so torej orodja, kot sta PHP in JavaScript, nepogrešljiva.

Literatura

[1] D. Flanagan. *JavaScript: The Definitive Guide, Sixth Edition*. O'Reilly Media, 2011.

[2] M. Pilgrim. *HTML5: Up and Running*. O'Reilly Media, 2010.

[3] T. B. Hansen, J. Lengstorf. *PHP for Absolute Beginners 2nd Edition*. Apress, 2014.

Internetni viri:

[4] Ajax definition. [Citirano 20.10.2014]
Dostopna na spletnem naslovu:
http://www.techterms.com/definition/ajax

[5] Cloud9. [Citirano 09.11.2014]
Dostopna na spletnem naslovu:
https://c9.io/

[6] Codeanywhere. [Citirano 09.11.2014]
Dostopna na spletnem naslovu:
https://codeanywhere.com/

[7] CodeMirror editor. [Citirano 21.10.2014]
Dostopna na spletnem naslovu:
http://codemirror.net/

[8] DigitalOcean wiki. [Citirano 20.11.2014]
Dostopna na spletnem naslovu:
http://en.wikipedia.org/wiki/DigitalOcean

[9] FileZilla wiki. [Citirano 20.11.2014]
Dostopna na spletnem naslovu:
http://en.wikipedia.org/wiki/FileZilla

[10] Google Chrome wiki. [Citirano 20.11.2014]
Dostopna na spletnem naslovu:
http://sl.wikipedia.org/wiki/Google_Chrome

[11] HTML Specifikacija. [Citirano 15.10.2014]
Dostopna na spletnem naslovu:
http://www.w3.org/TR/1999/REC-html401-19991224/intro/intro.html

[12] HTML5 Application Cache. [Citirano 12.11.2014]
Dostopna na spletnem naslovu:
http://www.w3schools.com/html/html5_app_cache.asp

[13] HTML5 Canvas. [Citirano 12.11.2014]
Dostopna na spletnem naslovu:
http://www.w3schools.com/html/html5_canvas.asp

[14] JavaScript tutorial. [Citirano 15.10.2014]
Dostopna na spletnem naslovu:
http://www.w3schools.com/js/

[15] JavaScript tutorial. Try it Yourself Example. [Citirano 04.11.2014]
Dostopna na spletnem naslovu:
http://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_body

[16] jQuery definition. [Citirano 18.10.2014]
Dostopna na spletnem naslovu:
http://www.techterms.com/definition/jquery

[17] Kaj je CSS? [Citirano 20.10.2014]
Dostopna na spletnem naslovu:
http://www.presentia.si/baza-znanja-helpdesk/2008/kaj-je-css/

[18] Knjižnica Downloadify. [Citirano 04.11.2014]
Dostopna na spletnem naslovu:
https://github.com/dcneiner/Downloadify

[19] Knjižnica jQuery tree. [Citirano 04.11.2014]
Dostopna na spletnem naslovu:
https://code.google.com/p/dwpe/source/browse/trunk/tree/js/jQuery.tree.js

[20] PHP 5 Tutorial. [Citirano 04.11.2014]
Dostopna na spletnem naslovu:
http://www.w3schools.com/php/

[21] Ponudnik gostovanja v oblaku DigitalOcean. [Citirano 13.11.2014]
Dostopna na spletnem naslovu:
https://www.digitalocean.com/

[22] PuTTY wiki. [Citirano 20.11.2014]
Dostopna na spletnem naslovu:
http://simple.wikipedia.org/wiki/PuTTY

[23] Skripti jezik PHP. [Citirano 15.10.2014].
dostopen na spletnem naslovu:
http://www.php.net/

[24] Sublime Text wiki. [Citirano 13.11.2014]
Dostopna na spletnem naslovu:
http://en.wikipedia.org/wiki/Sublime_Text